



VIEW

TABELAS VIRTUAIS EM SQL

LABORATÓRIO DE BANCO DE DADOS

VANESSA BORGES

Visão

- Visão é uma tabela simples derivada de outras tabelas
 - Uma view não necessariamente existe em forma física; ela é considerada uma **tabela virtual**, ao contrário das **tabelas da base**
- Forma de se especificar uma tabela que precisa ser acessada **frequentemente**, embora essa tabela não exista fisicamente
- Facilita a escrita de consultas complexas
- **Existem duas formas de um SGBD implementar visões:**
 - **Modificação de consultas:** a visão é criada a cada consulta
 - **Materialização de visões:** a visão é criada na primeira consulta





Visão – modificação de consulta

CREATE

- **Sintaxe para a criação de uma VIEW:**

```
CREATE [OR REPLACE] [ TEMP | TEMPORARY ] VIEW <nome_visão>  
[(nome_atributo [, nome_atributo ...])]  
AS <SELECT ...>;
```

- **Listar View no psql: \dv**
- **Onde:**
 - **OR REPLACE:** caso a VIEW já exista com o mesmo nome ela será substituída pela mais recente
 - **TEMP | TEMPORARY:** especificado quando queremos que a View seja temporária, as quais são eliminadas de forma automática quando a sessão atual é finalizada
 - **Lista de atributos:** opcional
 - **SELECT:** especifica o conteúdo da visão

<https://www.postgresql.org/docs/current/sql-createview.html>





Visão – modificação de consulta

DROP

- **Sintaxe para a remoção de uma VIEW:**

```
DROP VIEW [ IF EXISTS ] <nome_visão>  
[CASCADE | RESTRICT];
```

- Onde:
 - **IF EXISTS:** não retorna erro caso a visão não exista
 - **CASCADE:** remove automaticamente outras visões que dependem desta
 - **RESTRICT:** rejeita operação caso existam dependências



Operações sobre visões

- Visão “**somente leitura**”

- Visão que permite somente a realização de operações de seleção

- Visão “**atualizável**”

- Visão que permite as operações de seleção, inserção, remoção e atualização
 - seleção: SELECT
 - inserção: INSERT INTO
 - remoção: DELETE
 - atualização: UPDATE



Operações sobre visões

- Visões inerentemente **atualizáveis não possuem**
 - Operadores de conjunto
 - DISTINCT
 - Funções de agregação
 - GROUP BY
 - ORDER BY
 - Subconsulta aninhada
 - JOIN
 - Stored procedures



Problema de atualização da visão

- O problema de atualização por meio de visões é a ambiguidade na interpretação do comando
 - Exemplo:

```
CREATE VIEW vpessoa AS  
(SELECT cpf, pnome as nome, sexo FROM funcionario)  
UNION  
(SELECT fcpf, nome_dependente as nome, sexo FROM  
dependente);
```

```
INSERT INTO vpessoa VALUES ('123456789', 'Jose', 'M');
```

Em qual tabela base
será inserida a tupla?



Problema de atualização da visão

```
CREATE VIEW vtrabalhaem_sum_horas AS  
(SELECT fcpf, sum(horas) as qtd_horas FROM trabalha_em GROUP BY fcpf) ;
```

- Não há correspondência direta da soma de horas com um atributo da tabela base
- Essa visão não pode ser atualizável



Problema de atualização da visão

```
CREATE VIEW vfuncionario_unome AS  
(SELECT DISTINCT unome FROM funcionario);
```

```
UPDATE vfuncionario_unome  
SET unome='B.'  
WHERE unome = 'Borg';
```

- Cada tupla de vfuncionario_unome pode corresponder a várias tuplas de funcionário, então não há correspondência direta de um atributo da visão com um atributo da tabela base
- Portanto esta visão não pode ser atualizável



Visão – atualização

- Exemplo de atualização de view:

```
CREATE VIEW vprojeto5  
AS SELECT * FROM projeto WHERE  
projnumero>10;
```

```
UPDATE vprojeto5 SET projlocal='Stanford'  
WHERE projlocal='Houston';
```

Em geral, para ser atualizável, a visão deve ser derivada de apenas uma tabela base e deve conter a chave primária da tabela (projnumero)



Materialização de visões

- Como as Views **são apenas para leitura e representação lógica dos dados** que estão armazenados nas tabelas do banco de dados, podemos “materializá-las”, ou seja, armazená-las fisicamente no disco
- Discussão
 - Replicação dos dados
 - Armazenamento de dados agregados
 - Custo de consultas x custo de atualização

São muito utilizadas em aplicações onde os dados podem ficar temporariamente desatualizados, com atualizações periódicas, por exemplo, dados estatísticos, pois mantêm a vantagem de desempenho sem prejuízo na propagação de atualizações.





Visão – materialização de visões

CREATE

- Criar view materializada:

```
CREATE MATERIALIZED VIEW <nome_visão>  
[(nome_atributo [, nome_atributo ...])]  
AS <SELECT ...> [ WITH [ NO ] DATA ];
```

- Listar View no psql: \dm
- Onde:
 - WITH [NO] DATA: especifica se a visão deve ser preenchida no momento da criação. Caso contrário, a visualização materializada será marcada como não-digitalizável e não poderá ser consultada até que a opção REFRESH MATERIALIZED VIEW seja usada





Visão – materialização de visões

DROP

- Sintaxe para a remoção de uma VIEW:

```
DROP MATERIALIZED VIEW [ IF EXISTS ] <nome_visão>  
[CASCADE | RESTRICT];
```

- Onde:
 - **IF EXISTS**: não retorna erro caso a visão não exista
 - **CASCADE**: remove automaticamente outras visões que dependem desta
 - **RESTRICT**: rejeita operação caso existam dependências





Visão – materialização de visões

REFRESH

- Atualização de view materializada

```
REFRESH MATERIALIZED VIEW <nome_visão>;
```





Visão – materialização de visões

ALTER

- Renomeia uma visão já criada
- **Sintaxe para a alteração de uma VIEW:**

```
ALTER MATERIALIZED VIEW <nome_visão>  
RENAME TO <novo_nome_visão>;
```



Implementação de Visões

- **Existem duas formas de um SGBD implementar visões:**
 - **Modificação de consultas:** a visão é criada a cada consulta
 - **VANTAGEM:**
 - Não é necessário mecanismo de atualização para garantia de consistência da visão em relação às tabelas-base
 - **DESVANTAGEM:**
 - Desempenho de consultas frequentes é prejudicado
 - **Materialização de Visões:** a visão é criada na primeira consulta
 - **VANTAGEM:**
 - Consultas frequentes à visão têm bom desempenho
 - **DESVANTAGEM:**
 - Atualizações nas tabelas-base devem ser propagadas para as visões

